NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Technical Memorandum 33-539

An Organization of A Digital Subsystem for Generating Spacecraft Timing and Control Signals

M. Perlman

JET PROPULSION LABORATORY
CALIFORNIA INSTITUTE OF TECHNOLOGY
PASADENA, CALIFORNIA

May 15, 1972

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Technical Memorandum 33-539

An Organization of A Digital Subsystem for Generating Spacecraft Timing and Control Signals

M. Perlman

JET PROPULSION LABORATORY
CALIFORNIA INSTITUTE OF TECHNOLOGY
PASADENA, CALIFORNIA

May 15, 1972

Prepared Under Contract No. NAS 7-100
National Aeronautics and Space Administration

PREFACE

The work described in this report was performed by the Astrionics Division of the Jet Propulsion Laboratory.

CONTENTS

I. Introduction 1

II. Mathematical Background 2

A. Fundamental Theorem of Arithmetic 2

B. Greatest Common Divisor 2

C. Linear Diophantine Equations and Congruences 3

D. The Chinese Remainder Theorem for Integers 8

III. Application of the Chinese Remainder Theorem to Timing and Control Signal Generation 11

IV. Recurrence Relationships for the Modulo-p_i Counters 17

Appendix A. Algorithms for Finding the Greatest Common Divisor 24

Appendix B. An APL Program for the Chinese Remainder Theorem . . . 29

References 33

TABLES

1. A modulo-60 counter decomposed into parallel modulo $m_1 = 3$, modulo $m_2 = 4$, and modulo $m_3 = 5$ counters 19

FIGURES

1. Proposed spacecraft timing and control signal generation for Mariner Venus-Mercury 1973 20

2. State diagrams for modulo-2, -3, -5, and -7 FSR counters . . 21

3. A generalized FSR 22

4. Implementation of FSRs whose state diagrams appear in Fig. 2 23

B-1. An APL program for solving simultaneous congruences over pairwise relatively prime moduli (Chinese Remainder Theorem) 31

B-2. APL solution of example 7 32

ABSTRACT

A modulo-M counter (of clock pulses) is decomposed into parallel modulo-m_i counters, where each m_i is a prime power divisor of M. Each m_i is a cascade of α_i identical modulo-p_i counters, where $m_i = p_i^{\alpha_i}$. The modulo-p_i counters are feedback shift registers which cycle through p_i distinct states. By this organization, every possible nontrivial data frame subperiod (in terms of clock pulse intervals) and delayed subperiod may be derived.

The number of clock pulses required to bring every (or a subset of all) modulo-p_i counter to a respective designated state or count is determined by The Chinese Remainder Theorem. This corresponds to the solution of simultaneous congruences over relatively prime moduli.

I. INTRODUCTION

Each clock pulse interval (CPI) of a fixed-length serial data frame which is repetitive may be put into a one-to-one correspondence with the integers 0, 1, . . . , M - 1, where M is the frame length in terms of CPIs. Thus, a sequential network capable of assuming M distinct states can be used to time-tag each CPI. In effect, the sequential network is autonomous (i.e., has no inputs except clock pulses) and serves as a modulo-M counter (of clock pulses). When synthesizing synchronous sequential logic, the logical designer avoids races and hazards (Ref. 1) if loading and clocking frequency limitations of the digital elements are respected.

When M is not prime, every possible proper divisor (where divisors 1 and M are excluded) corresponds to a non-trivial subperiod of the periodic data frame. In effect, the finite-state machine (i.e., modulo-M counter) is decomposed into k submachines, where k is the number of proper divisors of M (Ref. 2).

Decomposition introduces flexibility which would be costly to achieve in a large modulo-M counter. The combinational logic required to translate a state to a count is comparable for the modulo-M counter and the decomposed machine. However, decomposition allows for subperiod and delayed subperiod generation with a modest amount of decoding logic. In a single machine (i.e., modulo-M counter), the decoding logic is prohibitive for subperiod and delayed subperiod generation unless the count code is fixed-weighted. Fixed-weighted code generation in a synchronous mode, however, requires interstage gating (combinational logic), which grows with the capacity of the counter.

Minimum overall complexity is realized with feedback shift registers forming submachines which operate synchronously and in series-parallel. FSR codes are nonweighted for every modulo (except 2).

II. MATHEMATICAL BACKGROUND

A. Fundamental Theorem of Arithmetic

A prime number is an integer $p > 1$ that is divisible by 1 and p only.

Every integer $M > 1$ may be uniquely expressed (except for order) as a prime or a product of two or more primes. The unique factorization of M into primes is known as the fundamental theorem of arithmetic (Ref. 3). That is,

$$M = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_n^{\alpha_n} \tag{1}$$

where the primes p_i are distinct and the exponents α_i are positive integers.

B. Greatest Common Divisor

The greatest common divisor (gcd) of two integers a and b is the largest integer d which divides a and b. This is denoted as

$$d = (a, b)$$

where d is the largest integer, such that

$$d \mid a \quad \text{and} \quad d \mid b$$

The expression $d \mid a$ (d divides a) means $a = dq$, where a, d and q are integers. Let

$$a = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_n^{\alpha_n}$$

$$b = p_1^{\beta_1} p_2^{\beta_2} \cdots p_n^{\beta_n}$$

where $\alpha_i \geq 0$ and $\beta_i \geq 0$. Then,

$$d = (a,b) = p_1^{\min(\alpha_1,\beta_1)} \; p_2^{\min(\alpha_2,\beta_2)} \; \cdots \; p_n^{\min(\alpha_n,\beta_n)}$$

Example 1

For

$$a = 588 = 2^2 \cdot 3 \cdot 7^2$$

$$b = 15,435 = 3^2 \cdot 5 \cdot 7^3$$

$$d = (a,b) = 3 \cdot 7^2 = 147$$

Generally a and b are not in factored form. The Euclidean algorithm (Appendix I) provides an efficient means for determining (a, b) without employing factorization.

A very important relationship exists between the integers a, b and d = (a, b), as stated in the following theorem (Ref. 3).

If d = (a, b), there exist integers x and y, such that

$$ax + by = d \tag{2}$$

An important consequence of the foregoing theorem is that if a and b are relatively prime, where (a, b) = 1, there exist integers x and y, such that

$$ax + by = 1 \tag{3}$$

Conversely, if a representation such as (3) exists for 1, then (a, b) = 1.

C. Linear Diophantine Equations and Congruences

Diophantine equations, named in honor of the Greek mathematician Diophantos, are equations in one or more variables whose solutions are

integers (or in some cases rational numbers). Equations (2) and (3) are examples of linear Diophantine equations. The solution to

$$ax + by = c$$

where a, b, and c are given integers and x and y are integers to be determined, involves finding an x such that ax and c yield the same remainder when divided by b. If a solution exists, then

$$b \mid (c - ax)$$

and y can be chosen as

$$y = \frac{c - ax}{b}$$

Suppose two integers s and t leave the same remainder r when divided by a third integer m. Then,

$$s = q_1 m + r$$

$$t = q_2 m + r$$

where $0 \leq r < m$ and $s - t = (q_1 - q_2) m$. It follows that

$$m \mid (s - t)$$

Gauss, the 19th century German mathematician, suggested the following "congruence" notation:

$$s \equiv t \bmod m$$

meaning s is congruent to t modulo m if $m \mid (s - t)$. The linear Diophantine equation

$$ax + by = c \qquad (4)$$

can be expressed as a linear congruence, namely

$$ax \equiv c \bmod b \tag{5}$$

Before considering the conditions for the existence of a solution or solutions to (4), or equivalently (5), let us state some additional properties of congruences.

If $a \equiv c \bmod b$ and $d \equiv e \bmod b$. Then,

(1) $a \pm d \equiv c \pm e \bmod b$

(2) $ad \equiv ce \bmod b$

also

(3) $ka \equiv kc \bmod b$ for every integer k

If $ka \equiv kc \bmod b$ and $(k, b) = d$, then

(4) $a \equiv c \bmod (b/d)$

Note that when $(k, b) = 1$, then (4) becomes

$$a \equiv c \bmod b$$

That is, the divisor k must be relatively prime to the modulo b in order to perform cancellation without altering the modulus.

Returning to (4), or equivalently (5), a necessary and sufficient condition for a solution or solutions to exist is that

$$(a, b) \mid c$$

The number of solutions which are incongruent modulo m (hence, distinct) is exactly (a, b).

When a and b are relatively prime, one and only one solution exists regardless of the integer value of c.

Example 2

$$1485x \equiv 15 \bmod 2795$$

First, (1485, 2795) is determined by the Euclidean algorithm shown in Appendix A.

$$2795 = 1485 \cdot 1 + 1310$$

$$1485 = 1310 \cdot 1 + 175$$

$$1310 = 175 \cdot 7 + 85$$

$$175 = 85 \cdot 2 + 15$$

$$85 = 15 \cdot 6 + 5$$

$$15 = 5 \cdot 3 + 0$$

Thus, (1485, 2795) = 5 and $5 \mid 15$. This indicates that there are five solutions for example 2.

From property (4) of congruences

$$\frac{1485}{5}x \equiv \frac{15}{5} \bmod \left(\frac{2,795}{(5, 2795)}\right)$$

$$297x \equiv 3 \bmod 559$$

The foregoing congruence has <u>one</u> solution, since (297, 559) = 1.

From the definition of congruences

$$297x - 559y = 3$$

$$x = \frac{559y + 3}{297} = y + \frac{262y + 3}{297}$$

$$\frac{262y + 3}{297} = s \text{ (an integer)}$$

$$262y = 297s - 3$$

$$y = s + \frac{35s - 3}{262}$$

$$\frac{35s - 3}{262} = t, \qquad 35s = 262t + 3$$

$$s = 7t + \frac{17t + 3}{35}$$

$$\frac{17t + 3}{35} = u, \qquad 17t = 35u - 3$$

$$t = 2u + \frac{u - 3}{17}$$

$$\frac{u - 3}{17} = w, \qquad u = 17w + 3$$

The smallest positive value of u occurs for w = 0. It follows from w = 0,

$$u = 3,\ t = 6,\ s = 45,\ y = 51$$

and x = 96 is the solution to

$$297x \equiv 3 \bmod 559$$

whereas

$$1485x \equiv 15 \bmod 2795$$

has <u>five</u> distinct (i.e., incongruent modulo-2795) solutions, namely,

$$96,\ 96 + 559,\ 96 + 2 \cdot 559,\ 96 + 3 \cdot 559,\ 96 + 4 \cdot 559$$

or 96, 655, 1214, 1773, and 2332. Given the general form

$$ax \equiv c \bmod b, \text{ where } (a, b) = d$$

A solution for x, say x_0, yields all d solutions, as follows:

$$x_0,\ x_0 + \frac{b}{d},\ x_0 + \frac{2b}{d}, \cdots,\ x_0 + \frac{(d-1)b}{d}$$

Corresponding to each solution for x is a solution for y in the linear Diophantine form

$$ax - by = c$$

However, as in example 2, values of y are often not required.

A more efficient method for finding a solution for x is given in Appendix I.

D. The Chinese Remainder Theorem for Integers

The Chinese Remainder Theorem guarantees a unique solution for simultaneous congruences over moduli which are relatively prime by pairs. The theorem may be stated as follows (Ref. 3):

Every system of linear congruences in which the moduli are relatively prime in pairs is solvable, the solution being unique modulo, the product of the moduli.

Given the simultaneous congruences

$$\begin{aligned} x &\equiv a_1 \bmod m_1 \\ x &\equiv a_2 \bmod m_2 \\ &\cdots \\ x &\equiv a_n \bmod m_n \end{aligned} \qquad (6)$$

where $(m_i, m_j) = 1$ for all i, j, where $i \neq j$ and $a_1, a_2, \cdots, a_n$ are any set of integers, let

$$M = m_1 m_2 \cdots m_n$$

and

$$M_i = \frac{M}{m_i}$$

Since $(M_i, m_i) = 1$, a unique solution exists for y_i in the linear congruence

$$M_i y_i \equiv 1 \bmod m_i \text{ for all } i$$

There is one and only one solution for x, which is determined as follows:

$$x \equiv \sum_{i=1}^{n} a_i y_i M_i \bmod M \tag{7}$$

Note that, as expressed in (7), x is a solution of each congruence in (6).

$$a_i y_i M_i \equiv a_i \bmod m_i$$

$$\equiv 0 \bmod m_j, \text{ where } j \neq i$$

The latter results since m_j is a factor of M_i. The value of x is such that $0 \leq x < M$.

Example 3

$x \equiv 1 \bmod 3 \quad a_1 = 1 \quad m_1 = 3$

$x \equiv 2 \bmod 4 \quad a_2 = 2 \quad m_2 = 4$

$x \equiv 3 \bmod 5 \quad a_3 = 3 \quad m_3 = 5$

$$M = 3 \cdot 4 \cdot 5 = 60$$

$$M_1 = 20,\ M_2 = 15,\ M_3 = 12$$

$$20y_1 \equiv 1 \bmod 3$$

$$15y_2 \equiv 1 \bmod 4$$

$$12y_3 \equiv 1 \bmod 5$$

Unique solutions for y_1, y_2 and y_3 are 2, 3 and 3, respectively.

$$x \equiv (40a_1 + 45a_2 + 36a_3) \bmod 60$$

$$x \equiv (40 \cdot 1 + 45 \cdot 2 + 36 \cdot 3) \bmod 60$$

$$x \equiv 58 \bmod 60$$

Check

$$58 \equiv 1 \bmod 3$$

$$58 \equiv 2 \bmod 4$$

$$58 \equiv 3 \bmod 5$$

A modulo-3, a modulo-4, and a modulo-5 counter would be in state 1 2 3 (i.e., $a_1 = 1$, $a_2 = 2$ and $a_3 = 3$) for $n = 58 + k60$ CPI, where $k = 0, 1, 2, \cdots$ (State 1 2 3 repeats every 60 CPIs; see Table 1.)

III. APPLICATION OF THE CHINESE REMAINDER THEOREM TO TIMING AND CONTROL SIGNAL GENERATION

The following data frame length has been proposed for the Mariner Venus-Mercury 1973 spacecraft:

$$M = 14,817,600$$

Unique factorization (except for order) gives

$$M = 2^6 3^3 5^2 7^3$$

Let

$$M = m_1 m_2 m_3 m_4$$

where

$$m_1 = 2^6 = 64$$

$$m_2 = 3^3 = 27$$

$$m_3 = 5^2 = 25$$

$$m_4 = 7^3 = 343$$

The arrangement of m_i factors is arbitrary as long as pairwise relative primeness holds. The unique factorization into products of powers of distinct primes, where each distinct prime power corresponds to an m_i, guarantees pairwise relative primeness. Furthermore, prime power factorization enables one to enumerate all the divisors of M. Let

$$p_1 = 2,\ p_2 = 3,\ p_3 = 5 \text{ and } p_4 = 7$$

The following polynomial factors, when multiplied, yield terms which correspond to every divisor of M:

$$(1 + p_1 + p_1^2 + p_1^3 + p_1^4 + p_1^5 + p_1^6)$$

$$\times (1 + p_2 + p_2^2 + p_2^3)(1 + p_3 + p_3^2)$$

$$\times (1 + p_4 + p_4^2 + p_4^3)$$

The number of terms in the resulting polynomial is

$$(6 + 1)(3 + 1)(2 + 1)(3 + 1) = 336$$

Thus, there are 336 divisors of M, 334 of which are proper (1 and M are improper).

As shown in Figure 1, there are four counters, which count in modulo m_1, modulo m_2, modulo m_3, and modulo m_4. Each is composed of identical cascaded feedback shift registers (FSRs), where the number of states a particular FSR cycles through is a prime factor of M. Each FSR is designed to operate synchronously. Furthermore, the successive states through which a modulo-p_i FSR cycles (except for $p_1 = 2$) is not in binary order. Unused states are always driven into the major cycle (see Ref. 2). An output corresponding to a scale of p_i is clocked to the next modulo-p_i counter to realize a modulo-p_i^2 count, etc.

As shown in Fig. 1, all m_i states associated with the modulo-m_i counter may be decoded. In practice, however, only those states corresponding to a count of interest are decoded.

Example 4

Assume that CPI 10,942 must be identified.

$$\begin{aligned} 10{,}942 &\equiv 62 \bmod 64 \\ &\equiv 7 \bmod 27 \\ &\equiv 17 \bmod 25 \\ &\equiv 309 \bmod 343 \end{aligned}$$

CPI 10,942 occurs when a_1, a_2, a_3, and a_4 are 62, 7, 17, and 309, respectively. The occurrence is once per frame. That is,

$$\text{CPI } x = 10,942 + kM \text{ for } k = 0, 1, \cdots$$

Example 5

CPI $x = 0 + k400$ for $k = 0, 1, \cdots$ is to be generated. That is, M is to be divided into equal subframes of length 400.

$$400 = 2^4 \cdot 5^2$$

Let

$$\hat{m}_1 = 2^4 = p_1^4$$

$$\hat{m}_2 = m_3 = 5^2 = p_3^2$$

$$\hat{M} = \hat{m}_1 \hat{m}_2 = 400$$

The content of the leftmost four stages of m_1 and all six stages of m_3 must be detected, so that when all zeros are stored, an output x is generated. The first output may be delayed up through $\hat{M} - 1$ CPIs (up to 399) by detecting an appropriate nonzero $\hat{a}_1 \hat{a}_2$ combination. Assume that the first output is to be delayed 72 CPIs.

$$72 \equiv 8 \bmod 16$$

$$\equiv 22 \bmod 25$$

The combination $\hat{a}_1 \hat{a}_2$ of 8 22 will appear at CPI $x = 72 + k400$, where

$$k = 0, 1, \cdots$$

Example 6

In example 5, assume that $\hat{a}_1 = 15$ and $\hat{a}_2 = 20$. The number of CPIs designated by x required for this $\hat{a}_1\hat{a}_2$ combination to appear can be determined by the Chinese Remainder Theorem.

$$x \equiv 15 \bmod 16$$

$$x \equiv 20 \bmod 25$$

$$\hat{M}_1 = \frac{\hat{M}}{\hat{m}_1} = \frac{400}{16} = 25$$

$$\hat{M}_2 = \frac{\hat{M}}{\hat{m}_2} = 16$$

$$25y_1 \equiv 1 \bmod 16$$

$$16y_2 \equiv 1 \bmod 25$$

Unique solutions for y_1 and y_2 are 9 and 11, respectively.

$$x \equiv (255\hat{a}_1 + 176\hat{a}_2) \bmod 400$$

$$x \equiv 95 + k400 \text{ for } k = 0, 1, \cdots$$

Note that

$$95 \equiv 15 \bmod 16$$

$$\equiv 20 \bmod 25$$

Thus, the $\hat{a}_1\hat{a}_2$ combination (i.e., count) of 15 20 will appear for the first time at CPI 95 and will reappear every 400 CPIs thereafter.

Example 7

Given

$$M = 2^6\, 3^3\, 5^2\, 7^3 = 14,817,600$$

where

$$m_1 = 2^6 = 64$$

$$m_2 = 3^3 = 27$$

$$m_3 = 5^2 = 25$$

$$m_4 = 7^3 = 343$$

and

$$M_1 = \frac{M}{m_1} = 231,525$$

$$M_2 = \frac{M}{m_2} = 548,800$$

$$M_3 = \frac{M}{m_3} = 592,704$$

$$M = \frac{M}{m_4} = 43,200$$

The minimum number of clock pulses required to realize the counts

$$a_1 = 37,\ a_2 = 17,\ a_3 = 2,\ a_4 = 341$$

is determined as follows:

A unique solution for each y_i is guaranteed, where

$$231,525\ y_1 \equiv 1 \bmod 64$$

$$548,800\ y_2 \equiv 1 \bmod 27$$

$$592,704\ y_3 \equiv 1 \bmod 25$$

$$43,200\ y_4 \equiv 1 \bmod 343$$

The solutions are

$$y_1 = 45,\ y_2 = 13,\ y_3 = y_4 = 19$$

and

$$x \equiv (M_1 y_1 a_1 + M_2 y_2 a_2 + M_3 y_3 a_3 + M_4 y_4 a_4) \bmod M$$

$$x = 9,039,077 + k14,817,600 \text{ for } k = 0, 1, 2, \cdots$$

The minimum number of clock pulses required is

$$x = 9,039,077 \text{ clock pulses}$$

Check:

$$\begin{aligned} 9,039,077 &\equiv 37 \bmod 64 \\ &\equiv 17 \bmod 27 \\ &\equiv 2 \bmod 25 \\ &\equiv 341 \bmod 343 \end{aligned}$$

IV. RECURRENCE RELATIONSHIPS FOR THE MODULO-p_i COUNTERS

An r-stage feedback shift register which provides a count modulo-p_i may be characterized by an rth-order recurrence relation (i.e., an rth-order difference equation).

$$b_k = f(b_{k-1}, b_{k-2}, \cdots, b_{k-r}) \tag{8}$$

The bit fed back at CPI k is denoted by b_k. The content of the ith stage at CPI k becomes the content of the (i + 1)th stage at CPI k + 1. That is,

$$b_{k-i} = b_{(k+1)-(i+1)} \tag{9}$$

Expression (9) accounts for the shifting in the register. From (8), the bit being fed back is a Boolean function of the contents of the register. The initial state of stage i is b_{-i}, where CPI k = 0. The number of stages required for a modulo-p_i counter is r, where r satisfies the following inequalities:

$$2^{r-1} < p_i \leq 2^r$$

The recurrence relationships for the proposed modulo-p_i counters in example 7 and Fig. 1, where $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, and $p_4 = 7$, are:

$$p_1 = 2 \qquad b_k = 1 \oplus b_{k-1} = b'_{k-1}$$

$$p_2 = 3 \qquad b_k = b'_{k-1} b'_{k-2}$$

$$p_3 = 5 \qquad b_k = b'_{k-2} b'_{k-3}$$

$$p_4 = 7 \qquad b_k = b_{k-1} b'_{k-2} b_{k-3} + b'_{k-1} b'_{k-3} \tag{10}$$

The symbols $\oplus$, ', and + denote exclusive—or complementation and inclusive—or (logical summation), respectively. Juxtaposition denotes logical multiplication. The initial state of each stage is assumed to be 0 (i.e., $b_{-i} = 0$ for all i).

State diagrams of each of the four FSRs appear in Fig. 2. Note that unused states are always driven into the desired cycle of states.

The FSRs appearing in Fig. 1 are not shown in detail. In particular, the feedback networks are not explicitly drawn. See Fig. 3 for a generalized FSR (without a decoding network). The state of the ith stage at clock pulse interval (CPI) is denoted by b_{k-i}. The binary digit being fed back at CPI k is denoted by b_k.

The content of the first or leftmost stage is replenished by b_k after shifting, where b_k is a Boolean function of $b_{k-1}, b_{k-2}, \cdots, b_{k-r+1}, b_{k-r}$. The right-hand side of the recurrence relations in (10) corresponds to Boolean feedback functions.

FSR implementations for (10) appear in Fig. 4. The shift register stages are 1-enable JK flip-flops, whose characteristic equation is

$$Q = Jq' + K'q$$

where J and K are 1-enable inputs and q and Q are the present and next state, respectively.

Symbolism in Fig. 4 has been simplified, using the following correspondences:

$$b_{k-i} \leftrightarrow b_i \text{ and } b_k \leftrightarrow b$$

$$J_{k-i} \leftrightarrow J_i$$

$$K_{k-i} \leftrightarrow K_i$$

Table 1. A modulo-60 counter decomposed into parallel modulo $m_1 = 3$, modulo $m_2 = 4$, and modulo $m_3 = 5$ counters

CPI n	Count state			CPI n	Count state		
	m_1	m_2	m_3		m_1	m_2	m_3
0	0	0	0	30	0	2	0
1	1	1	1	31	1	3	1
2	2	2	2	32	2	0	2
3	0	3	3	33	0	1	3
4	1	0	4	34	1	2	4
5	2	1	0	35	2	3	0
6	0	2	1	36	0	0	1
7	1	3	2	37	1	1	2
8	2	0	3	38	2	2	3
9	0	1	4	39	0	3	4
10	1	2	0	40	1	0	0
11	2	3	1	41	2	1	1
12	0	0	2	42	0	2	2
13	1	1	3	43	1	3	3
14	2	2	4	44	2	0	4
15	0	3	0	45	0	1	0
16	1	0	1	46	1	2	1
17	2	1	2	47	2	3	2
18	0	2	3	48	0	0	3
19	1	3	4	49	1	1	4
20	2	0	0	50	2	2	0
21	0	1	1	51	0	3	1
22	1	2	2	52	1	0	2
23	2	3	3	53	2	1	3
24	0	0	4	54	0	2	4
25	1	1	0	55	1	3	0
26	2	2	1	56	2	0	1
27	0	3	2	57	0	1	2
28	1	0	3	58	1	2	3
29	2	1	4	59	2	3	4

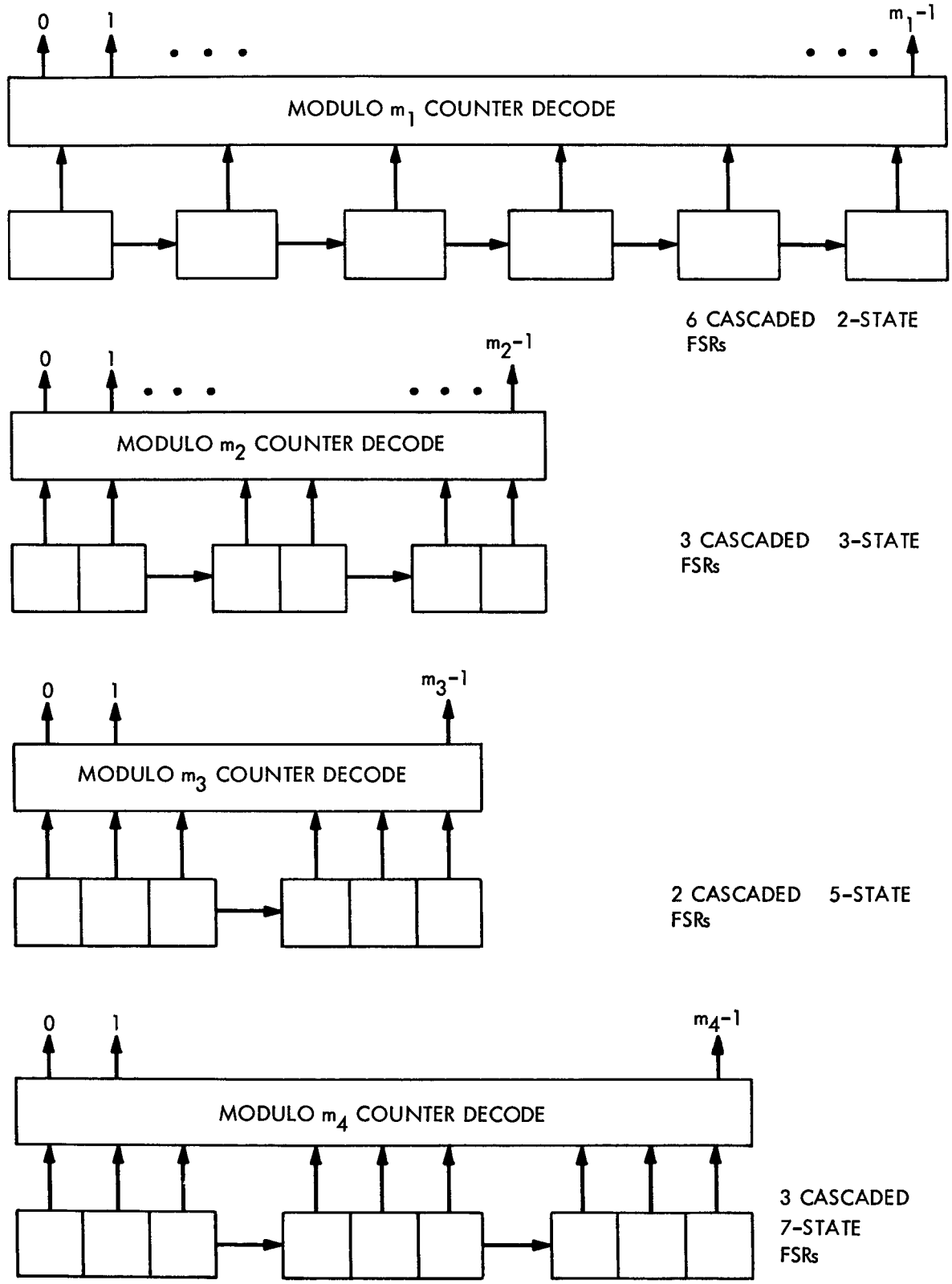

Fig. 1. Proposed spacecraft timing and control signal generation for Mariner Venus-Mercury 1973

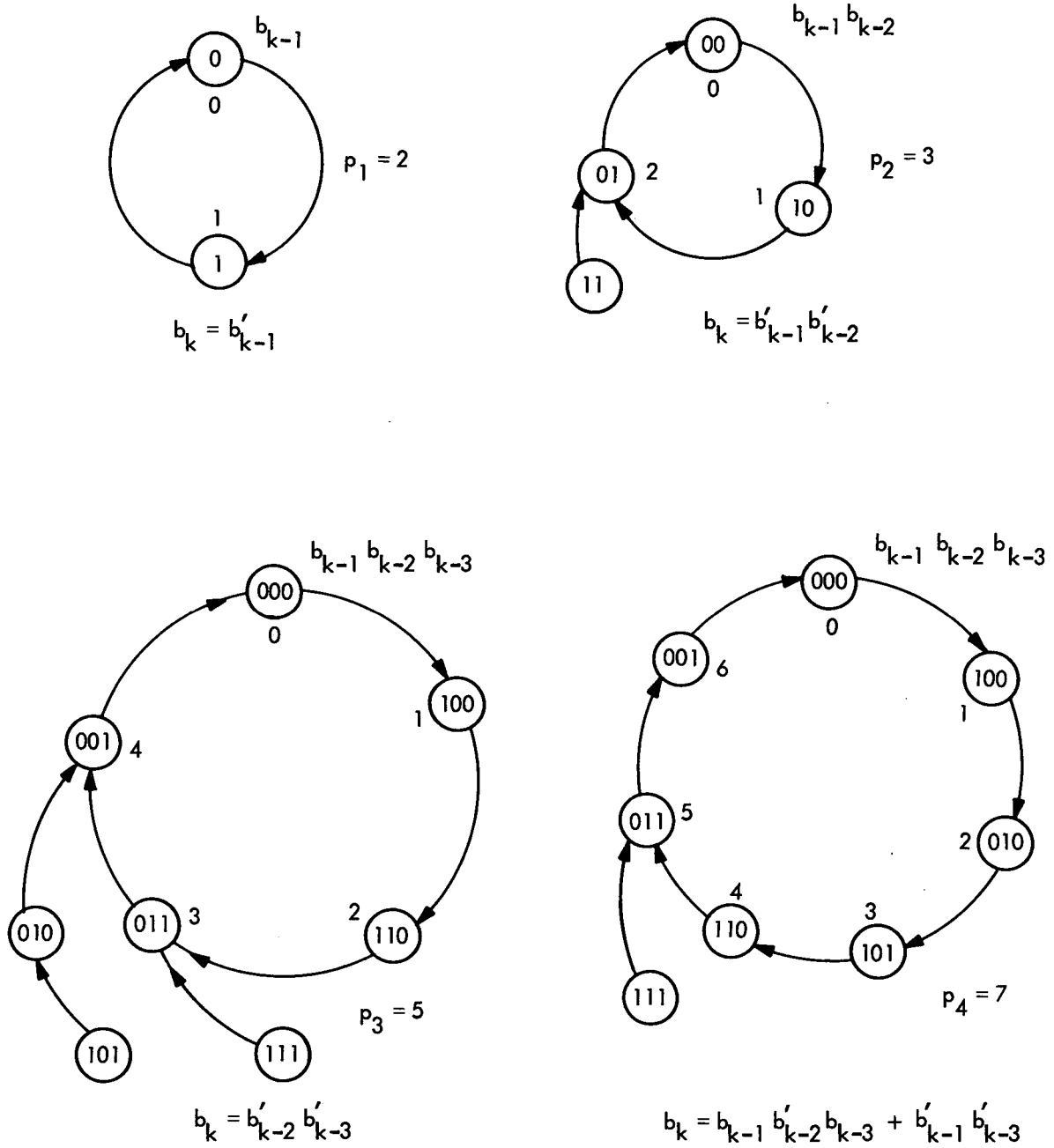

Fig. 2. State diagrams for modulo-2, -3, -5, and -7 FSR counters

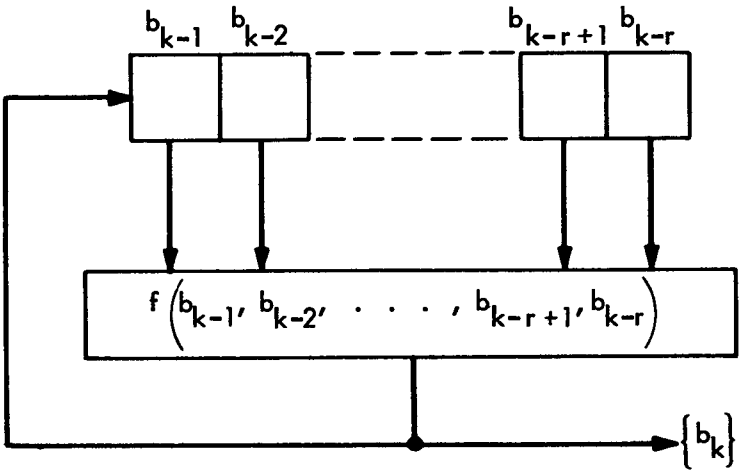

Fig. 3. A generalized FSR

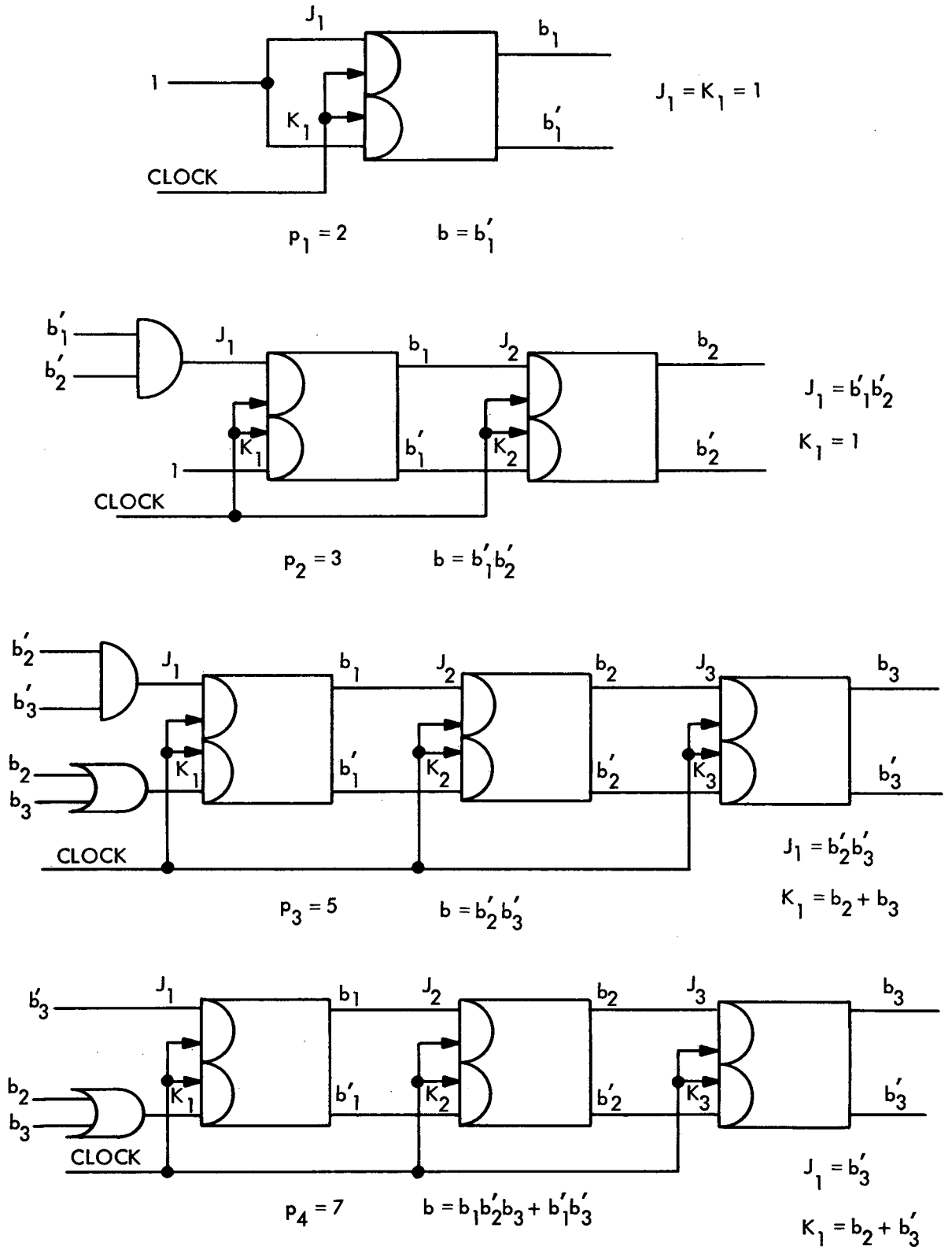

Fig. 4. Implementation of FSRs whose state diagrams appear in Fig. 2

APPENDIX A. ALGORITHMS FOR FINDING THE GREATEST COMMON DIVISOR

I. THE EUCLIDEAN ALGORITHM

The greatest common divisor of two integers may be computed by means of the Euclidean Algorithm (Refs. 3 and 4).

Let r_{-2} and r_{-1} denote two positive integers, where

$$r_{-2} > r_{-1} > 0$$

$$\begin{aligned}
r_{-2} &= r_{-1}h_0 + r_0 && 0 < r_0 < r_{-1} \\
r_{-1} &= r_0 h_1 + r_1 && 0 < r_1 < r_0 \\
&\vdots && \vdots \\
r_k &= r_{k+1}h_{k+2} + r_{k+2} && 0 < r_{k+2} < r_{k+1} \\
&\vdots && \vdots \\
r_{n-3} &= r_{n-2}h_{n-1} + r_{n-1} && 0 < r_{n-1} < r_{n-2} \\
r_{n-2} &= r_{n-1}h_n + 0 && 0 = r_n < r_{n-1}
\end{aligned} \qquad (11)$$

Note that successive values of r are decreasing integers, so that $r_n = 0$ for some integer n. Also

$$(r_{-2}, r_{-1}) = (r_{-1}, r_0) = (r_0, r_1) = \cdots = (r_{n-2}, r_{n-1}) = (r_{n-1}, 0) = r_{n-1}$$

Linear Diophantine equations, and hence linear congruences, can be solved by means of the Euclidean Algorithm. This can best be illustrated by an example.

Example 8

The linear congruence in example 7,

$$231,525\ y_1 \equiv 1 \bmod 64$$

was asserted to have the solution $y_1 = 45$. The congruence may be expressed as a linear Diophantine equation as follows:

$$231,525\ y_1 - 64z = 1$$

where y_1 and z are integers. First, the Euclidean Algorithm is employed to determine (231525, 64).

$$\begin{aligned} 231,525 &= 64 \cdot 3617 + 37 \\ 64 &= 37 \cdot 1 + 27 \\ 37 &= 27 \cdot 1 + 10 \\ 27 &= 10 \cdot 2 + 7 \\ 10 &= 7 \cdot 1 + 3 \\ 7 &= 3 \cdot 2 + 1 \\ 3 &= 1 \cdot 3 \end{aligned}$$

and (231525, 64) = 1. Note that $h_0 = 3617$, $h_1 = 1$, $h_2 = 1$, $h_3 = 2$, $h_4 = 1$, and $h_5 = h_{n-1} = 2$. Also, $r_5 = r_{n=1} = 1 = (231525, 64)$. Using successive partial results of the Euclidean Algorithm in reverse yields

$$\begin{aligned} 1 &= 7 - 3 \cdot 2 \\ &= 7 - (10 - 7 \cdot 1)2 = 7 \cdot 3 - 10 \cdot 2 \\ &= (27 - 10 \cdot 2)3 - 10 \cdot 2 = 27 \cdot 3 - 10 \cdot 8 \\ &= 27 \cdot 3 - (37 - 27 \cdot 1)8 = 27 \cdot 11 - 37 \cdot 8 \\ &= (64 - 37 \cdot 1)11 - 37 \cdot 8 = 64 \cdot 11 - 37 \cdot 19 \\ &= 64 \cdot 11 - (231,525 - 64 \cdot 3617)19 \\ &= 64 \cdot 68,734 - 231,525 \cdot 19 \\ &= 231,525\,(-19) + 64 \cdot 68,734 \end{aligned}$$

Thus,

$$y_1 = -19 \equiv 45 \bmod 64$$

and

$$z = 68,734$$

The foregoing method is inefficient, since all the partial results of the Euclidean Algorithm must be stored in order to solve y_1 and z. Furthermore, in solving a linear congruence, the value of z (i.e., the multiple of the modulo) is not required.

II. SUCCESSIVE CONVERGENTS

The Euclidean Algorithm, together with two additional recurrence relationships, may be used to solve linear Diophantine equations. This method does not require the storage of all the partial results in the Euclidean Algorithm. Given

$$r_{-2} > r_{-1} > 0$$

$$p_{-2} = 0 \qquad p_{-1} = 1$$

$$q_{-2} = 1 \qquad q_{-1} = 0$$

These integer values are the initial conditions of the following respective recurrence relationships, where k = 0:

$$r_{k-2} = r_{k-1}h_k + r_k \qquad 0 \le r_k < r_{k-1}$$

$$p_k = p_{k-1}h_k + p_{k-2} \tag{12}$$

$$q_k = q_{k-1}h_k + q_{k-2} \tag{13}$$

The first expression is the Euclidean Algorithm, from which successive values of h_k are determined. In turn, successive values of p_k and q_k are determined; the process is terminated at n, where $r_n = 0$. The final values are h_{n-1}, p_{n-1}, and q_{n-1}, and the linear Diophantine equation has the form

$$r_{-1}p_{n-1} - r_{-2}q_{n-1} = (-1)^n(r_{-2}, r_{-1}) \tag{14}$$

The quotients p_k/q_k are known as successive convergents of the continued fraction form of the Euclidean Algorithm (Refs. 3 and 4).

Returning to example 8, h_0 through $h_5 = h_{n-1}$ were computed in accordance with (11). Concurrently, (12) and (13) can be applied to determine p_5 and q_5, as follows:

$$p_0 = p_{-1}h_0 + p_{-2} = 1 \cdot 3617 + 0 = 3617$$

$$q_0 = q_{-1}h_0 + q_{-2} = 0 \cdot 3617 + 1 = 1$$

$$p_1 = p_0h_1 + p_{-1} = 3617 \cdot 1 + 1 = 3618$$

$$q_1 = q_0h_1 + q_{-1} = 1 \cdot 1 + 0 = 1$$

$$p_2 = p_1h_2 + p_0 = 3618 \cdot 1 + 3617 = 7235$$

$$q_2 = q_1h_2 + q_0 = 1 \cdot 1 + 1 = 2$$

$$p_3 = p_2h_3 + p_1 = 7235 \cdot 2 + 3618 = 18,088$$

$$q_3 = q_2h_3 + q_1 = 2 \cdot 2 + 1 = 5$$

$$p_4 = q_3h_4 + q_2 = 5 \cdot 1 + 2 = 7$$

$$p_5 = p_4h_5 + p_3 = 25,323 \cdot 2 + 18,088 = 68,734$$

$$q_5 = q_4h_5 + q_3 = 7 \cdot 2 + 5 = 19$$

Since n - 1 = 5, the process terminates and (14) can be evaluated.

$$r_{-1}p_5 - r_{-2}q_5 = (-1)^6 (r_{-2}, r_{-1})$$

$$64 \cdot 68,734 - 231,525 \cdot 19 = 1$$

or

$$231,525\ (-19) + 64 \cdot 68,734 = 1$$

The results agree with those in example 8, where

$$y_1 = -19 \equiv 45 \bmod 64, \text{ and } z = 68,734$$

Note that only r_{k-2}, r_{k-1}, h_k, q_{k-2}, and q_{k-1} need be stored (current values). Values of p_k do not have to be computed, since $p_{n-1} = z$, which is not needed in evaluating y_1.

Whenever the values of the m_i are large (i.e., exceed several hundred), the preceding method is most efficient in terms of iterations (time) and storage required by a general-purpose computer.

APPENDIX B. AN APL PROGRAM FOR THE CHINESE REMAINDER THEOREM

I. AN APL PROGRAM FOR SOLVING SIMULTANEOUS CONGRUENCES OVER PAIRWISE RELATIVELY PRIME MODULI

APL (A Programming Language) is an interactive programming language created by K. E. Iverson. Complex sequential processes may be concisely described in APL with a minimum amount of self-training. It is particularly suited for testing feasibility of algorithms without extensive programming experience. Manipulations on entire arrays of operands can be efficiently performed in APL (Ref. 5).

An APL terminal at the Jet Propulsion Laboratory connects via telephone data lines to a time-shared IBM 360 model 50 general-purpose computer. The work-space capacity is 48K words.

The statements comprising an APL program entitled "PRIMECOUNT" appear in Fig. B-1. Upon the request of the program, the user enters the number of counters (i. e., number of congruences to be solved simultaneously), each modulus, the count associated with each modulus (the a_i associated with the m_i in Eq. 6, and the clock frequency. The program determines each y_i needed in Eq. 7 to determine the number of clock pulses x in order to achieve the desired count in each counter simultaneously, the minimum number of clock pulses (x reduced modulo M), and the time in milliseconds required to generate the minimum number of clock pulses. Time in seconds is computed to eight significant digits.

Figure B-2 is the APL program solution of example 7, with a clock frequency of 20 MHz assumed. The boxlike symbols followed by a colon are points in time where the user is requested to enter parameters. Note that the solution of any number of simultaneous congruences over pairwise relatively prime moduli may be determined by the APL program providing work-space capacity is not exceeded and CPU time is acceptable (i. e., cost).

II. COMPUTATION OF y_i VALUES

Moduli whose values do not exceed several hundred are anticipated in future timing and control designs. Therefore, the iterative relations given in Appendix I were not used in the PRIMECOUNT APL program to determine

the values of y_i. Instead, y_i, starting with a value of 0, is incremented and tested to determine whether or not it satisfies

$$M_i y_i \equiv 1 \bmod m_i$$

That is, does

$$m_i \mid M_i y_i - 1 \tag{15}$$

where m_i and M_i are given and y_i is set at 1? If not, y_i is incremented to 2, and the test is repeated, etc. The loop may have to be traversed as many as $m_i - 1$ times before a y_i satisfying (15) is found. For large values of m_i, the Euclidean Algorithm for determining successive h values and the q_k recurrence relation for determining successive q values are recommended, as shown in Appendix A. For example 7, the APL program performs 45 iterations to determine y_1, whereas six iterations on h_k (h_0 through h_5) and six on q_k (q_0 through q_5) are required to determine the same y_1 in example 8.

Note that for a given timing and control system organization, the y_i are calculated once. Any number of sets of a_i may be entered. Statement [28] in the APL program in Fig. B-1 asks whether there are any additional counts. The user types a yes or no. A yes causes a branch to statement [20], preparing the program to accept a new set of a_i (i.e., counts). A no terminates the program, as shown in Fig. B-2.

```
      ∇PRIMECOUNT[⎕]∇
    ∇ PRIMECOUNT;N;M;Y;R;L;J;I;A;F;T
[1]   'ENTER THE NUMBER OF COUNTERS'
[2]   N←⎕
[3]   'ENTER EACH MODULUS'
[4]   M←⎕
[5]   →ERROR×⍳N≠⍴,M
[6]   Y←N⍴0
[7]   R←N⍴0
[8]   L←×/M
[9]   I←0
[10]  I←I+1
[11]  R[I]←L÷M[I]
[12]  J←0
[13]  J←J+1
[14]  →13×⍳1≠M[I]|J×R[I]
[15]  Y[I]←J
[16]  →10×⍳N≠I
[17]  'CORRESPONDING Y VALUES  ';Y
[18]  'ENTER CLOCK FREQUENCY IN MEGAHERTZ'
[19]  F←⎕
[20] BR1:'ENTER COUNT ASSOCIATED WITH EACH MODULUS'
[21]  A←⎕
[22]  →ERROR×⍳N≠⍴,M
[23]  X←L|+/L|A×L|Y×R
[24]  'CLOCK PULSES REQUIRED X=';X;'+';L;'×K FOR K=0,1,2,.....'
[25]  'MINIMUM X=';X
[26]  T←(⌊100000000×5E¯9+X÷F×1000000)÷100000
[27]  'MINIMUM X CORRESPONDS TO ';T;' MILLISECONDS'
[28]  'ADDITIONAL COUNTS?'
[29]  →BR1×⍳'Y'∊⍞
[30]  →0
[31] ERROR:'INVALID INPUT'
    ∇
```

Fig. B-1. An APL program for solving simultaneous congruences over pairwise relatively prime moduli (Chinese Remainder Theorem)

```
      PRIMECOUNT
ENTER THE NUMBER OF COUNTERS
□:
      4
ENTER EACH MODULUS
□:
      64 27 25 343
CORRESPONDING Y VALUES  45  13  19  19
ENTER CLOCK FREQUENCY IN MEGAHERTZ
□:
      20
ENTER COUNT ASSOCIATED WITH EACH MODULUS
□:
      37 17 2 341
CLOCK PULSES REQUIRED X=9039077+14817600×K FOR K=0,1,2,.....
MINIMUM X=9039077
MINIMUM X CORRESPONDS TO 451.95385 MILLISECONDS
ADDITIONAL COUNTS?
NO
```

Fig. B-2. APL solution of example 7

REFERENCES

1. McCluskey, E. J., Introduction to the Theory of Switching Circuits, McGraw Hill Book Co., New York, 1965.

2. Perlman, M., "Derivation of Timing and Control Signals for the Ultraviolet Spectrometer Data Automation System," Supporting Research and Advanced Development, Space Programs Summary 37-32, Vol. IV, pp. 188-195, Jet Propulsion Laboratory, Pasadena, California, April 30, 1965.

3. LeVeque, W. J., Topics in Number Theory, Vol. 1, Addison-Wesley Publishing Co., Reading, Massachusetts, 1956.

4. Berlekamp, E. R., Algebraic Coding Theory, McGraw Hill Book Co., New York, 1968.

5. Hellerman, H., Digital Computer Systems Principles, McGraw Hill Book Co., New York, 1967.

NASA — JPL — Coml., L.A., Calif.